PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Michael Freed; Elango Ganesan	Confirmation No.	4136
Serial No.:	09/900,494		
Filed:	July 6, 2001	Customer No.:	28863
Examiner:	Aravind K. Moorthy		
Group Art Unit:	2131		
Docket No.:	1014-064US01/JNP-0261		
Title:	LOAD BALANCING SECURE SOCKETS LAYER ACCELERATOR		

CERTIFICATE UNDER 37 CFR 1.8 I hereby certify that this correspondence is being transmitted via the United States Patent and Trademark Office electronic filing system on July 10, 2007.

By: Caryl Harriman
Name: Caryl Harriman

REPLY BRIEF

Mail Stop Appeal Brief – Patents
Board of Patent Appeals and Interferences
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313

Dear Sir:

This is a Reply Brief from the Examiner's Answer mailed May 10, 2007. The Notice of Appeal was filed on August 23, 2006 and Appellants' Appeal Brief was filed December 20, 2006.

No fee is believed due. Please charge any deficiencies or credits to Deposit Account No. 50-1778

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party in interest is Juniper Networks, Inc. of Sunnyvale, California.

RELATED APPEALS AND INTERFERENCES

There are no related appeals and interferences.

STATUS OF CLAIMS

Claims 1–28 are on appeal in this case. Claims 1–28 are rejected under 35 U.S.C. 112, first paragraph, asserting that the specification fails to describe the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention.

Claims 1–7 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. (USPN 6,799,202) ("Hankinson") in view of Toporek et al. (USPN 6,654,344) ("Toporek").

Claims 12–15, 17–21, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Abjanic (USPN 6,732,175) in view of Toporek.

Claims 25–28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Baskey (USPN 6,732,269) in view of Toporek.

STATUS OF AMENDMENTS

No amendments have been filed subsequent to the Final Office Action mailed April 20, 2006 from which this Appeal has been made.

SUMMARY OF THE CLAIMED SUBJECT MATTER

A concise summary of independent claim 1, 12 and 25 is provided within Appellant's Brief filed August 23, 2006. Per MPEP 1208, the Summary of the Claimed Subject Matter has been omitted from this Reply Brief.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The first grounds for rejection to be reviewed on Appeal is the rejection of claims 1–28 under 35 U.S.C. 112, first paragraph, asserting that the specification fails to describe the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention.

The second grounds for rejection to be reviewed on Appeal is the rejection of claims 1–7 and 22 under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. (USPN 6,799,202) ("Hankinson") in view of Toporek et al. (USPN 6,654,344) ("Toporek").

The third grounds for rejection to be reviewed on Appeal is the rejection of claims 12–15, 17–21, 23 and 24 under 35 U.S.C. 103(a) as being unpatentable over Abjanic (USPN 6,732,175) in view of Toporek.

The fourth grounds for rejection to be reviewed on Appeal is the rejection of claims 25–28 under 35 U.S.C. 103(a) as being unpatentable over Baskey (USPN 6,732,269) in view of Toporek.

ARGUMENTS

The First Ground of Rejection

Claims 1–28 are on appeal in this case. All claims are rejected under 35 U.S.C. 112, first paragraph, asserting that the specification fails to describe the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention. Appellants submit that, with respect to the first grounds of rejection, claims 1–28 stand or fall together.

Response to the Examiner's Argument

Claim 1 recites a load balancing acceleration device. With respect to claim 1, the central issue for the rejection under 35 U.S.C. 112, first paragraph is the requirement that the device include a decryption engine and a load balancing engine that "bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack." Similarly, with respect to claims 12 and 25, the claim elements at issue are "without processing the data packets with an application layer of a network stack."

In the Examiner's Answer, the Examiner based his entire argument on the premise that secure socket layer (SSL) encryption and decryption always occurs at the application layer. Specifically, with respect to the rejection of claim 1, the Examiner stated:

> *The examiner acknowledges that page 10 of the present application specifically states that "rather than transmitting packets up and down the TCP/IP stack as show in figure 2B, [the SSL accelerator of Figure 3] will perform the SSL encryption and decryption at the packet level before forwarding the packet on to its destination." The examiner points out to the board specifically in that statement that SSL encryption and decryption is being performed.* ***The examiner would also like to point out to the board that one of ordinary skill in the art knows that SSL is performed in the application layer. Therefore, if SSL encryption and decryption is being performed, it has to be accomplished in the application layer.***
>
> *The Appellants continue to argue (on page 14 of the appeal brief) that there are three modes that are described in great detail of pages 13-30 of the present application, including description of detailed flow charts showing how*

> *these various modes implement SSL within the accelerator without requiring that the application data be reassembled at the application layer.*
>
> *The examiner asserts that all three modes are described by the Appellants in the present application incorporate SSL.* ***The examiner emphasizes that SSL occurs in the application layer.*** *The examiner asserts that the Appellants have not provided any details in the specification how SSL is being performed outside of the application layer. (Ex. Ans., pp. 16-17) (emphasis added).*

Based on this premise, the Examiner ignored the teaching of the present application and concluded that Appellants' claimed load-balancing acceleration device having a decryption engine and load balancing engine that bypass an application layer of a network stack is unsupported. The Examiner's position is erroneous for several reasons.

First, Appellants respectfully disagree with the Examiner's position that SSL inherently always occurs at the application layer. The present application describes quite clearly that conventional client or server devices may implement SSL in the *session layer* (i.e., one layer below the application layer) to provide secure communication *sessions* between client devices and server devices. The present application describes applications executing within the application layer as producing outbound application-layer data that is passed down to the session layer for encryption via SSL for the communication session. Inbound SSL communications are decrypted *at the session layer* and then passed up to the application layer. For example, the present application clearly states:

> Figure 2B illustrates how SSL functions in the Open Systems Interconnect (OSI) Reference Model and in typical accelerators. The web client transmits data to the accelerator 250 in an encrypted form to the secure port 443 of the accelerator. In the client, the application layer protocol hands unencrypted data to the <u>session layer</u>; SSL encrypts the data and hands it down through the layers to the network IP layer, and on to the physical layers (now shown). Normally, a server will receive the encrypted data and when the server receives the data at the other end, it passes it up through the layers to the <u>session layer where SSL decrypts it and hands it off to the application layer (HTTP)</u>. The same happens in the typical SSL accelerator within the accelerator, where the data is handed to the application layer, processed, then returned down the stack from the HTTP layer to the IP layer for transmission to port 80 (in the clear) on the server coupled to the SSL accelerator. Once at the server, the data returns up the stack for processing in the application layer. Since the client and the SSL device have gone through the key negotiation handshake, the symmetric key used by SSL is the same at both ends.

> In essence, the HTTP packet must travel through the TCP stack four times, creating a latency and CPU overhead and requiring full TCP stack support in the accelerator . . . .[1]

This description provided by the present application is consistent with the industry-standard seven-layer Open Systems Interconnection (OSI) Basic Reference Model. As shown in the evidence attached hereto in Appendix E, the OSI model defines a layered, abstract description for communications and computer network protocol design, developed as part of Open Systems Interconnection (OSI) initiative. It is also called the OSI seven layer model. The layers, described below, are, from top to bottom, Application, Presentation, Session, Transport, Network, Data Link and Physical. The evidence attached hereto clearly shows that SSL can be incorporated within the session layer or even the presentation layer, i.e., not the application layer. In other words, contrary to the Examiner's unsupported statement, SSL is not inherently always performed within the application layer. Such an unsupported statement cannot be used as a basis for ignoring the teachings of Appellant's specification and rejecting Appellants' claims as non-enabled under 35 U.S.C. 112, first paragraph.

Second, the present application specifically describes techniques by which an acceleration device can perform encryption and decryption of data <u>on a packet level</u> and forward the data without handing off reassembled application-layer data up and down the TCP/IP stack to and from the application layer. To be clear, Appellants' claimed SSL acceleration device intercepts and processes SSL data, but this processing does <u>not</u> occur at the application layer, as argued by the Examiner. Moreover, unlike prior art acceleration devices, Appellants' claimed device does not require the application layer be invoked after SSL processing when forwarding the data. Appellants' specification describes techniques by which the intercepted SSL data can be processed *and forwarded* <u>at the packet level</u> without requiring complete reassembly of the application-layer data, thus allowing the application layer to be bypassed. This process is shown quite clearly shown in Figure 3 of the present application, reproduced and described at length in Appellants' Appeal Brief, pg. 13.

---

[1] Present application, pg. 5, ln. 16–pg. 6, ln. 6 (emphasis added).

Moreover, the present application further describes in detail three distinct modes supported by Appellants' SSL acceleration device. These three modes are described in great detail on pp. 13–30 (eighteen pages) of the present application, including description of detailed flowcharts showing how these various modes implement SSL within the accelerator without requiring that application data be completely reassembled and passed up the entire network stack. As just one example of the detailed technical disclosure within the present application, pg. 17, ll. 3–28 describes how SSL encrypted data can be processed at the packet level by utilizing a database and a buffer for processing TCP segments:

> The SSL accelerator 250 includes a TCP/SSL session database to track all communication sessions occurring through it. Each session will have one or more records associated with it, with each record comprising an association of the TCP session sequence and the SSL sequence. Hence, on receiving the initial SYN from client 100 at step 202, the SSL accelerator will create a database entry for the particular session, associating the TCP-SSL sequence number pairs. The data may be considered as a table, with each row in the table representing one entry in a given session. Hence, for each session, a typical record might include up to about 8 – 16 records, which include a TCP sequence number, SSL session number, an initialization vector (for DES and 3DES) and an expected ACK.
>
> During decryption, the device may utilize portions of its memory to buffer segments as necessary for decryption. The number and size of the buffers will depend on the cipher scheme used and the configuration of the packets, as well as whether the packets contain application data spanning multiple packets, referred to herein as multi-segment packets (and illustrated with respect to Figure 8). The SSL device can allocate SSL buffers as necessary for TCP segments. If, for example, application data having a length of 3000 bytes is transmitted via TCP segments having a length of 100 bytes, the device can, copy TCP segment 1 to a first SSL buffer, and start a timer, wait for packet 2 and when received, copy it to an SSL buffer and restart the timer, and finally when packet 3 is received, the SSL accelerator will copy it, decrypt all application data, authenticate it and forward the data on in the clear. (An alternative, bufferless approach is described below).

This section describes an example of how the SSL accelerator uses a TCP/SSL session database to track sessions, and TCP segments are buffered directly within SSL buffers and decrypted. Notably, an application-layer protocol, such as HTTP, is avoided.

Furthermore, the present application expressly recognizes that implementing encryption and decryption within an intermediate acceleration device at the packet level (rather than using application layer data at the application layer) may result in certain problems. For example, the present application recognizes that SSL segments may span

multiple TCP segments, which may result in problems if encryption and decryption occurs at the packet level as described by the present application. The present application, on pp. 25-29, then details certain solutions to those problems so that the claimed acceleration device can perform encryption and decryption at the packet level. As one example, the present application describes a unique bufferless or small buffer approach to handle the multisegment problem that may arise when SSL is implemented at the packet level by bypassing the application layer. In the bufferless approach, the present application describes the acceleration device as decrypting individual segments of multisegment SSL records, <u>but not authenticated</u> prior to being sent to the server. Only upon receipt of the packet carrying the last segment in the series is the data authenticated, however, individual SSL segments are not. This is but one example as to how the present application enables packet-level encryption and decryption that bypasses the application layer of the network stack. For the convenience of the Board, the solutions provided by the present application on pp. 25-29 are reproduced in full below:

> *There are numerous types of communications problems which may occur at various stages of data transfer between the SSL Accelerator, the client and the server. Some examples of these problems, and how the SSL device handles them, are set forth below. However, it will be understood that the number and type of errors which are possible in this sequence, and their attendant solutions, are too numerous to detail here.*
>
> *One type of problem is lost packets. Most lost packet cases can be recovered through use of the data structure mentioned above. As the data structure maintains the TCP sequence number, SSL sequence number, expected ACK and DES's Initialization vector, the SSL Accelerator device can roll back the SSL number to the previous TCP number received.*
>
> ***A different problem occurs not packets are lost, but when there is an SSL segmentation problem. Segmentation problems may occur when, for example, 1 SSL record spans over 3 TCP segments, i.e.: where SSL length=3000, and the TCP packet's length = 1000. This segmentation issue is illustrated in Figure 8. In this case, the Accelerator device cannot decrypt and authenticate the packet, since the MAC algorithm data will not arrive for another two segments.***
>
> *If, in the method of the invention, the accelerator uses a memory buffer, (as described above with respect to Figure 5) the Accelerator can allocate an SSL buffer for 3000 bytes, copy TCP segment 1 to the SSL buffer, and start a timer. When packet SSL/TCP packet 2 is received, it will be copied to an SSL buffer and the timer restarted. Then when packet 3 is received, the SSL accelerator will copy it, decrypt it, allocate 3 TCP, segments, and copy HTTP data into it. This may then be forwarded on in the clear.*

An alternative embodiment of the present invention utilizes a bufferless or small buffer approach to handle the multisegment problem. In the bufferless approach, individual segments of multisegment SSL records are decrypted, but not authenticated prior to being sent to the server. *Upon receipt of the last segment in the series (packet 3 in the above example), the data will be authenticated, however, individual segments are not. This greatly reduces the hardware requirements of the device be requiring little or no buffer memory allocated to multi segment SSL packets. For non-block ciphers, such as RC2 and RC4, this decryption can be performed on the fly. However, for block ciphers such as 3DES/DES, some buffering must occur. This is due to the fact that data for these ciphers must be combined from blocks. In these cases, only part of the data is decrypted and the rest is moved to the next segment. Hence, if there are more than two segments, and the encryption cipher is DES, with 8 byte blocks, the SSL device will buffer up to 7 bytes with additional 7 bytes sequentially moved until the last segment, with the last segment always having enough room to accommodate the data without breaking the server's MSS. In an exemplary design, the operational modes are configurable by a user so that the sacrifice of whether to potentially compromise security by not authenticating each packet is the user's choice. Nevertheless, because for block ciphers it is impossible to know the padding length before decryption is finished and the padding length is used to start calculating authentication, then authentication of the data in the multi-segment SSL data does occur upon receipt of the last segment - and the receipt of the MAC algorithm data and one is required to store all decrypted data into a buffer. If, however, the data cannot be authenticated at that time, the SSL device will send a reset to the server and an ALERT to the client, indicating a problem with the session has occurred and notifying the user. For block ciphers, the system does some buffering, but this minimal buffering will reduce latency.*

Another issue may occur when a "small" window problem occurs. Normally, communications between the Sever to Client occur as shown in Table 1:

TABLE 1

Client	*SSL Accelerator*	*Server*
		←TCP80 1=0
	encrypt *←SSL TCP443 1=0*	
		← TCP80 2=1000
	Encrypt *← SSL TCP443 2=1000*	
		← TCP80 3=2000
	Encrypt *← SSL TCP443 3=2000*	
TCP443 ACK=3000 →		
	TCP80 ACK=3000 →	

The small window problem may occur when, for example, the ServerMSS=1000, but Client understands an MSS=900. In this situation, if the client sends an ACK W=3000, the SSL accelerator will understand it is going to receive 3, 1000 byte segments. This problem is illustrated in Table 3. In Table 3, the server's packet length is, for example, 100 bytes. So instead of receiving 3, 1000 byte segments, the SSL accelerator will receive 30, 100 byte segments from the server. Once the SSL accelerator adds the SSL overhead, which in this example is 100 bytes, the packet size to be returned to the client doubles for each packet from the server:

TABLE 2

Client	*SSL Accelerator*	*Server*
Ack W=3000 -->		
	Ack W=2700 (SSL expecting 3 1000Segments)	
		<-- TCP 1=0, l=100
	Encrypt *<-- SSL TCP 1=0, l=200*	
		<-- TCP 2=100, l =100
	Encrypt *<-- SSL TCP 2=200, l =200*	
		<-- TCP 3=200, l=100
	Encrypt *<-- SSL TCP 3=400, l = 200*	
	□ □ □	
		<-- TCP 14=1400, l=100
	Encrypt *<-- SSL TCP 4=2800, l = 200*	
		<-- TCP 15=1500, l=100
	Encrypt *<-- SSL TCP 5=3000, l = 200*	
		<-- TCP 16=1600, l=100

The SSL accelerator cannot send TCP packet 16 because client's window is full already (with 15, 200 byte packets).

In this case, the SSL accelerator will buffer the Server's responses, starting from this point so that when a next TCP ACK=3000 is received from the client, the SSL accelerator will take the server response (packet 16) from the buffer, encrypt it and return it to the client.

If one of the foregoing problems occurs when the SSL accelerator is in a mode which does not support that particular type of communication, the SSL accelerator may switch modes to enable that type of communication to be handled.

For at least these reasons, one of ordinary skill would appreciate that the present application enables an acceleration device having a decryption engine and a load balancing engine that "bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack." Similarly, with respect to claims 12 and 25, the present application enables an SSL acceleration device that performs SSL encryption and operation, and selects a destination server "without processing the data packets with an application layer of a network stack."

Appellants' specification describes the claimed subject matter in such a way to enable one skilled in the art to make and use the invention. The Examiner's position that SSL must inherently be performed at the application layer ignores Appellants' specific teachings of techniques for processing SSL data with an intermediate device at the packet level without requiring that the data be reassembled and handed-off to the application layer. The rejection under 35 USC 112, first paragraph, should be withdrawn.

The Second Ground of Rejection

Claims 1–7 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. (USPN 6,799,202) ("Hankinson") in view of Toporek et al. (USPN 6,654,344) ("Toporek"). Claim 1 requires an acceleration device in which a decryption engine and the load balancing engine bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.

Response to the Examiner's Argument

In the Examiner's Answer, the Examiner pointed to no teaching in the prior art of an acceleration device in which a decryption engine and the load balancing engine bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack. Instead, the Examiner argued that the cited references teach load balancing (Ex. Ans., pg, 18) and that bypassing the application layer for load balancing would inherently result in increased speed (*Id.*, pg. 20). This analysis overlooks the requirement that Appellants' claimed acceleration device must be capable of decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.

As set forth in Appellants' Appeal Brief, even in combination, Hankinson in view of Toporek fail to provide any teaching relevant to providing encryption and decryption of secure data within an acceleration device in a manner that bypasses the application

layer. Toporek describes a mechanism for controlling data flow from a satellite. The portion of Toporek cited by the Examiner describes a satellite gateway that merely *relays* information between a client and a server. The related information is not processed by the satellite gateway other than to control the rate of flow through the link. Toporek indeed states that the relayed information can flow through the satellite gateway at the network layer and bypass the transport and application layers, but this is merely because the data is not processed whatsoever. There is no teaching in Hankinson in view of Toporek as to how a device could decrypt secure data using a process that bypasses the application layer. Even if the Examiner's characterization of Toporek is correct that Toporek "allows the network layer to communicate directly to the physical layer," the combination of references still provides no teaching as to how the application-layer (which is above both the network layer and the physical layer) could be avoided for functions like SSL that traditionally require the application-layer.[2]

More specifically, Appellants' claim 1 requires decrypting the data from the secure communication sessions of the clients without processing the data with the application layer of the network stack. Hankinson describes a distributed operating system and makes only a passing reference to SSL, and the Toporek device only *relays* information. This combination provides no teaching whatsoever as to how a secure client communication could be decrypted, such as SSL. In fact, the combination does not suggest that SSL would be processed any differently whatsoever. According to Toporek, satellite communications that need not be processed can be relayed without processing. According to Hankinson, network components can be distributed to multiple devices.

The Examiner's conclusion of obviousness is based entirely on the unsupported assumption that the Hankinson in view of Toporek could somehow achieve an acceleration device capable of decrypting secure client communications without processing the secure communication at the application layer. It is important for the Examiner to appreciate that SSL and other secure communications of application data encrypt blocks of application data to form secure records above the packet level. Once formed, the secure records are passed down the network stack to the network layer where

[2] Office Action, pg. 5, where the Examiner cites Toporek at col. 11, ll. 22–33.

the secure records are split into packets. Decrypting SSL records without processing the records at an application layer is a non-trivial problem that is not remotely answered or suggested by any of the references. As one example, handling SSL communications of encrypted application data may require reassembly of secure records of application data that span multiple packets. Techniques for addressing these and other issues associated with secure client communications without processing the packets at the application layer are described throughout the application. See, e.g., pp. 17 and 26, which describe techniques by which Appellants' acceleration device is able to decrypt SSL records that span multiple packets without processing the packets at the application layer.

With respect to encryption, Hankinson merely notes that the system may support SSL. Hankinson provides no suggestion that SSL is supported in any manner that is different from the prior art. Toporek provides no mention of SSL or decrypting secure communications whatsoever. Therefore, there is no reasonable expectation that the Hankinson web server could be successfully modified in view of Toporek so as to somehow incorporate the function of decrypting data from a communication session without processing the data at the application layer. There is a significant gap in the teachings of the references as to how such a feature may even be implemented. The fact that Hankinson makes a passing reference to SSL and that Toporek describes a mechanism for <u>relaying</u> packet information without processing the packets whatsoever provides no enabling description of how encrypted communications could be decrypted with an acceleration device without processing the secure data at the application layer.

The Examiner's suggestion that the Hankinson operating system that load balances processing task could somehow be modified in view of Toporek to load balance decrypted data without processing the decrypted data at the application layer again glosses over significant gaps in the teachings of the references.

The Third Ground of Rejection

Claims 12–15, 17–21, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Abjanic (USPN 6,732,175) in view of Toporek. Claim 12 requires, without processing the data packets with an application layer of a network stack, selecting

with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients.

Response to the Examiner's Arguments

The Examiner argues that Abjanic is not an application-layer content-based apparatus (Ex. Ans., pp. 21). This, however, is in direct contrast with the disclosure of Abjanic that clearly states that the network apparatus includes a content-based message director (e.g., *XML* director) to route or direct messages received from the network to one of the processing nodes *based upon the application data*, including business transaction information.[3] In direct contrast to the elements of claim 21, Abjanic makes clear that the described apparatus is an application-layer content-based switching apparatus. Abjanic provides numerous examples of how application layer data (XML data in this case) is extracted using the HTTP protocol (which is an application-layer protocol) in order to make forwarding decisions. Below is one brief example:

> *The application data is provided after the HTTP header, and in this example is provided as XML data. . . . [T]he present invention is directed to a technique to perform switching at a network apparatus based upon the application data, such as XML data (which includes business transaction information).*[4]

HTTP headers and XML data are only available at the application layer upon reassembly of application-layer data. Abjanic fails to teach or suggest mechanisms by which a load balancing acceleration device can, without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients.

In response, the Examiner also justifies the rejection based on the fact that Abjanic describes an SSL accelerator (Ex. Ans., pp. 21). The Examiner also argues that Abjanic includes a traffic manager that performs acceleration and load balancing based on

---

[3] Abjanic at Summary.
[4] Abjanic at col. 6, ll. 1–25.

business transactions, purchase orders, stock quotes or stock trades, and other "business transaction information." (Ex. Ans., pp., 22). This information, however, is application-layer information and would require reassembly of application-layer data. Certainly, business transaction information would not be available below the application layer. For this reason, Abjanic specifically states that application-layer data is required for the content-based switching.[5]

Thus, Abjanic provides no suggestion of performing SSL encryption and decryption, in any fashion different from the prior art, i.e., at the session layer and handing off reassembled application-layer data to the application layer. In fact, quite the contrary, Abjanic expressly requires application-layer data. Therefore, the combination of Abjanic in view of Toporek fails to provide any teaching for decrypting data packets of the secure protocol with the acceleration device to provide decrypted packet data, and without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients, as required by claim 12.

<u>The Fourth Ground of Rejection</u>

Claims 25–28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Baskey (USPN 6,732,269) in view of Toporek. Claim 25 requires a client device, a plurality of server devices, and an intermediate device coupled between the client devices and the server devices. Claim 25 further requires that the intermediate device intercepts a request from the client device for a secure communication session, and in response to the request, the intermediate device establishes a secure communication session with the client device, selects one of the server devices based on resource loading experienced by the server devices without processing the request with an application layer of a network stack, and establishes a non-secure communication session with the selected server device.

[5] Abjanic at col. 6, ll. 1–25.

Response to the Examiner's Arguments

The Examiner's basis for the rejection of claims 25-28 appears to be based squarely on the premise addressed above with respect to claim 1, i.e., that SSL must always be performed within the application layer. For example, the Examiner argues that Baskey describes two different connections that utilize an SSL proxy server (Ex. Ans., pg. 24). As discussed in Appellants' Appeal Brief, the Baskey approach explicitly requires that secure data travel two full networking stacks, including the application layer, and Baskey fails to describe any other mechanism for implementing the SSL proxy. The satellite relay function of Toporek, where packets are not even processed, provides no enabling teaching whatsoever as to how the Baskey device could be modified so that SSL functions could still be implemented yet in a manner that avoids the application layer. Nothing in Baskey or Toporek bridges this significant gap as to how the Toporek packet relay function that bypasses both the transport and the application layer could possibly be used within a load-balancing, full proxy device, such as the Baskey device.

For at least these reasons, the references fail to establish a prima facie case for non-patentability of Appellants' claims under 35 U.S.C. 103(a). Withdrawal of this rejection is requested.

Conclusion of Arguments

The Examiner erred in rejecting Appellants' claims under 35 U.S.C. 112, first paragraph and 35 U.S.C. 103(a). Reversal of these rejections and allowance of the pending claims are requested.

Respectfully submitted,

Date:
July 10, 2007

By: Kent J. Sieffert
Name: Kent J. Sieffert
Reg. No.: 41,312

Shumaker & Sieffert, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: (651) 735-1100 ext. 341
Facsimile: (651) 735-1102

APPENDIX I: CLAIMS ON APPEAL

Claim 1 (Previously Presented): A load balancing acceleration device, comprising:

a processor, memory and communications interface;

a TCP communications manager capable of interacting with a plurality of client devices and server devices simultaneously via the communications interface;

a secure communications manager to negotiate a secure communication session with one of the client devices;

an encryption and decryption engine instructing the processor to decrypt data received via the secure communication session and direct the decrypted data to one of said server devices via a second communication session; and

a load balancing engine associating each of said client devices with a respective one of said server devices based on calculated processing loads of each said server devices,

wherein the decryption engine and the load balancing engine bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.

Claim 2 (Previously Presented): The device of claim 1 wherein the TCP communications manager provides an IP address of an enterprise to said secure communications manager, and each of said plurality of servers devices is associated with the enterprise.

Claim 3 (Previously Presented): The device of claim 2 wherein the secure communications manager negotiates a secure communication session with each of said plurality of client devices over an open network.

Claim 4 (Previously Presented): The device of claim 3 wherein the TCP communications manager negotiates a separate, open communications session with one of the plurality of servers devices associated with the enterprise for each secure communications session negotiated with the ~~a~~ client devices based on the associations of said client devices to said server devices by said load balancing engines.

Claim 5 (Previously Presented): The device of claim 1 wherein the encryption and decryption engine decrypts the data on a packet level by decrypting packet data received on the communications interface via the secure communications session to extract a secure record, decrypting application data from the secure record in the packet data, and outputting the decrypted application data from the secure record to the one of said server devices via the second communication session without processing the application data with ~~an~~ the application layer of the network stack.

Claim 6 (Previously Presented): The device of claim 5 wherein the load-balancing engine selects the second communication session.

Claim 7 (Previously Presented): The device of claim 1 wherein the TCP communications manager responds to TCP communications negotiations directly for an enterprise.

Claim 8 (Previously Presented): The device of claim 1,

wherein the TCP communications manager receives packets from the client devices, and

wherein the TCP communications manager changes destination IP addresses for the packets to IP addresses for the server devices.

Claim 9 (Previously Presented): The device of claim 8,

wherein the TCP communications manager maintains TCP communication sessions with the server devices, and

wherein the secure communications manager negotiates a secure communication session for each TCP communications session.

Claim 10 (Original): The device of claim 9 wherein the secure communications manager responds to all secure communications with each client device.

Claim 11 (Previously Presented): The device of claim 9 wherein the secure communications manager changes a destination IP address for each packet to a server IP address.

Claim 12 (Previously Presented): A method for performing acceleration of data communications between a plurality of customer devices attempting to communicate with an enterprise having a plurality of servers, comprising:

providing an intermediate acceleration device enabled for secure communication with the customer devices, wherein the acceleration device has an IP address associated with the enterprise;

receiving with the acceleration device communications directed to the enterprise in a secure protocol from one of the customer devices;

decrypting data packets of the secure protocol with the acceleration device to provide decrypted packet data;

without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients; and

forwarding the decrypted packet data from the acceleration device to the selected server of the enterprise.

Claim 13 (Original): The method of claim 12 further including the steps of:

receiving application data from the selected server of the enterprise;

encrypting the application data received from the selected server; and

forwarding encrypted application data to the customer device.

Claim 14 (Previously Presented): The method of claim 12 wherein the step of receiving communications directed to the enterprise includes receiving with the device communications having a destination IP address of the enterprise.

Claim 15 (Previously Presented): The method of claim 12 further including the step of negotiating the secure protocol session with the customer device by responding as the enterprise to the customer devices.

Claim 16 (Previously Presented): The method of claim 12 further wherein the step of forwarding comprises:

modifying a destination IP address of data packets from an IP address associated with the enterprise IP to an IP address for the selected server.

Claim 17 (Previously Presented): The method of claim 12 wherein the step of forwarding comprises:

establishing an open communication session from the acceleration device to the selected server, and

mapping the decrypted packet data to the open communication session established with the selected server.

Claim 18 (Previously Presented): The method of claim 17 wherein the open communication session is established via a secure network.

Claim 19 (Previously Presented): The method of claim 12 wherein the step of receiving comprises:

receiving encrypted data having a length greater than a TCP segment carrying said data; and

wherein said step of decrypting comprises:

buffering the encrypted data in a memory buffer in the acceleration device, the buffer having a length equivalent to the block cipher size necessary to perform the cipher; and

decrypting the buffered segment of the received encrypted data to provide decrypted application data.

Claim 20 (Previously Presented): The method of claim 19 further including the step of authenticating the data on receipt of a final TCP segment on a packet level without processing the application data with the ~~an~~ application layer of the network ~~a TCP/IP~~ stack.

Claim 21 (Original): The method of claim 19 further including the step of generating an alert if said step of authenticating results in a failure.

Claim 22 (Previously Presented): The device of claim 1, wherein the device comprises a network router.

Claim 23 (Previously Presented): The method of claim 12, wherein decrypting data packets comprises decrypting the data packets at a packet level of the network ~~a TCP/IP~~ stack.

Claim 24 (Previously Presented): The method of claim 12, wherein decrypting data packets comprises:

decrypting the data packets to extract a secure record,

decrypting application data from the secure record, and

authenticating the application data without processing the application data with ~~an~~ the application layer of the network stack.

Claim 25 (Previously Presented): A system comprising:

a client device;

a plurality of server devices; and

an intermediate device coupled between the client devices and the server devices,

wherein the intermediate device intercepts a request from the client device for a secure communication session, and

wherein, in response to the request, the intermediate device establishes a secure communication session with the client device, selects one of the server devices based on resource loading experienced by the server devices without processing the request with an application layer of a network stack, and establishes a non-secure communication session with the selected server device.

Claim 26 (Previously Presented): The system of claim 25, wherein the intermediate device receives encrypted data from the client device via the secure communication session, decrypts the data and forwards the decrypted data to the selected server device via the non-secure communication session.

Claim 27 (Previously Presented): The system of claim 25, wherein the intermediate device receives unencrypted data from the selected server device via the non-secure communication session, encrypts the data and forwards the encrypted data to the client device via the secure communication session.

Claim 28 (Previously Presented): The system of claim 25, wherein the intermediate device comprises a network router.

APPENDIX II: EVIDENCE

1. **OSI Reference Model — The ISO Model of Architecture for Open Systems InterconnectionPDF (776 KiB), Hubert Zimmermann, IEEE Transactions on Communications, vol. 28, no. 4, April 1980, pp. 425 - 432.**

2. **OSI model. (2007, July 7). In Wikipedia, The Free Encyclopedia. Retrieved 18:36, July 10, 2007, from http://en.wikipedia.org/w/index.php?title=OSI_model&oldid=143155382.**

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-28, NO. 4, APRIL 1980 425

OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection

HUBERT ZIMMERMANN

(Invited Paper)

Abstract—Considering the urgency of the need for standards which would allow constitution of heterogeneous computer networks, ISO created a new subcommittee for "Open Systems Interconnection" (ISO/TC97/SC16) in 1977. The first priority of subcommittee 16 was to develop an architecture for open systems interconnection which could serve as a framework for the definition of standard protocols. As a result of 18 months of studies and discussions, SC16 adopted a layered architecture comprising seven layers (Physical, Data Link, Network, Transport, Session, Presentation, and Application). In July 1979 the specifications of this architecture, established by SC16, were passed under the name of "OSI Reference Model" to Technical Committee 97 "Data Processing" along with recommendations to start officially, on this basis, a set of protocols standardization projects to cover the most urgent needs. These recommendations were adopted by TC97 at the end of 1979 as the basis for the following development of standards for Open Systems Interconnection within ISO. The OSI Reference Model was also recognized by CCITT Rapporteur's Group on "Layered Model for Public Data Network Services."

This paper presents the model of architecture for Open Systems Interconnection developed by SC16. Some indications are also given on the initial set of protocols which will likely be developed in this OSI Reference Model.

I. INTRODUCTION

IN 1977, the International Organization for Standardization (ISO) recognized the special and urgent need for standards for heterogeneous informatic networks and decided to create a new subcommittee (SC16) for "Open Systems Interconnection."

The initial development of computer networks had been fostered by experimental networks such as ARPANET [1] or CYCLADES [2], immediately followed by computer manufacturers [3], [4]. While experimental networks were conceived as heterogeneous from the very beginning, each manufacturer developed his own set of conventions for interconnecting his own equipments, referring to these as his "network architecture."

The universal need for interconnecting systems from different manufacturers rapidly became apparent [5], leading ISO to decide for the creation of SC16 with the objective to come up with standards required for "Open Systems Interconnection." The term "open" was chosen to emphasize the fact that by conforming to those international standards, a system will be open to all other systems obeying the same standards throughout the world.

The first meeting of SC16 was held in March 1978, and initial discussions revealed [6] that a consensus could be reached rapidly on a layered architecture which would satisfy most requirements of Open Systems Interconnection with the capacity of being expanded later to meet new requirements. SC16 decided to give the highest priority to the development of a standard Model of Architecture which would constitute the framework for the development of standard protocols. After less than 18 months of discussions, this task was completed, and the ISO Model of Architecture called the Reference Model of Open Systems Interconnection [7] was transmitted by SC16 to its parent Technical Committee on "Data Processing" (TC97) along with recommendations to officially start a number of projects for developing on this basis an initial set of standard protocols for Open Systems Interconnection. These recommendations were adopted by TC97 at the end of 1979 as the basis for following development of standards for Open Systems Interconnection within ISO. The OSI Reference Model was also recognized by CCITT Rapporteur's Group on Public Data Network Services.

The present paper describes the OSI Architecture Model as it has been transmitted to TC97. Sections II-V introduce concepts of a layered architecture, along with the associated vocabulary defined by SC16. Specific use of those concepts in the OSI seven layers architecture are then presented in Section VI. Finally, some indications on the likely development of OSI standard protocols are given in Section VII.

Note on an "Interconnection Architecture"

The basic objective of SC16 is to standardize the rules of interaction between interconnected systems. Thus, only the external behavior of Open Systems must conform to OSI Architecture, while the internal organization and functioning of each individual Open System is out of the scope of OSI standards since these are not visible from other systems with which it is interconnected [8].

It should be noted that the same principle of restricted visibility is used in any manufacturer's network architecture in order to permit interconnection of systems with different structures within the same network.

These considerations lead SC16 to prefer the term of "Open Systems Interconnection Architecture" (OSIA) to the term of "Open Systems Architecture" which had been used previously and was felt to be possibly misleading. However, for unclear reasons, SC16 finally selected the title "Reference Model of Open Systems Interconnection" to refer to this Interconnection Architecture.

Manuscript received August 5, 1979; revised January 16, 1980.
The author is with IRIA/Laboria, Rocquencourt, France.

0090-6778/80/0400-0425$00.75 © 1980 IEEE

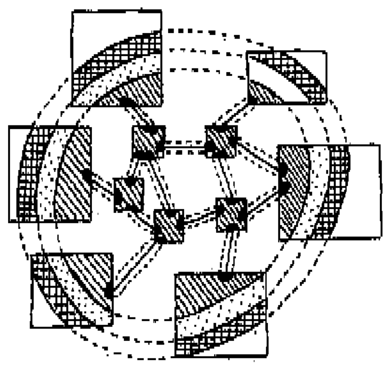


Fig. 1. Network layering.

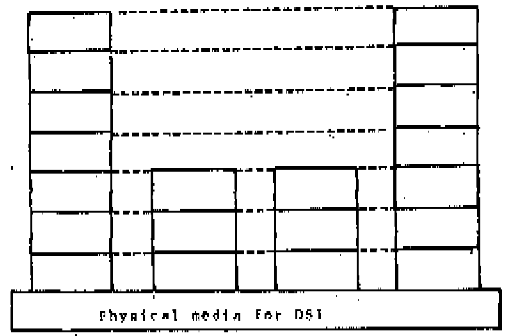


Fig. 2. An example of OSI representation of layering.

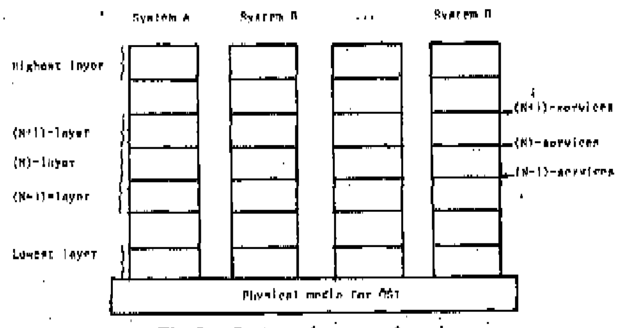


Fig. 3. Systems, layers, and services.

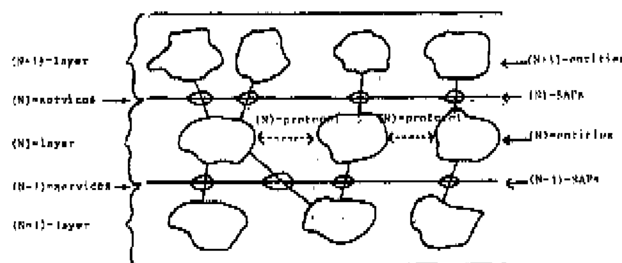


Fig. 4. Entities, service access points (SAP's), and protocols.

II. GENERAL PRINCIPLES OF LAYERING

Layering is a structuring technique which permits the network of Open Systems to be viewed as logically composed of a succession of layers, each wrapping the lower layers and isolating them from the higher layers, as exemplified in Fig. 1.

An alternative but equivalent illustration of layering, used in particular by SC16 is given in Fig. 2 where successive layers are represented in a vertical sequence, with the physical media for Open Systems Interconnection at the bottom.

Each individual system itself is viewed as being logically composed of a succession of subsystems, each corresponding to the intersection of the system with a layer. In other words, a layer is viewed as being logically composed of subsystems of the same rank of all interconnected systems. Each subsystem is, in turn, viewed as being made of one or several entities. In other words, each layer is made of entities, each of which belongs to one system. Entities in the same layer are termed *peer* entities.

For simplicity, any layer is referred to as the (N) *layer*, while its next lower and next higher layers are referred to as the $(N-1)$ layer and the $(N+1)$ layer, respectively. The same notation is used to designate all concepts relating to layers, e.g., entities in the (N) layer are termed (N) *entities*, as illustrated in Figs. 3 and 4.

The basic idea of layering is that each layer adds value to services provided by the set of lower layers in such a way that the highest layer is offered the set of services needed to run distributed applications. Layering thus divides the total problem into smaller pieces. Another basic principle of layering is to ensure independence of each layer by defining services provided by a layer to the next higher layer, independent of how these services are performed. This permits changes to be made in the way a layer or a set of layers operate, provided they still offer the same service to the next higher layer. (A more comprehensive list of criteria for layering is given in Section VI.) This technique is similar to the one used in structured programming where only the functions performed by a module (and not its internal functioning) are known by its users.

Except for the highest layer which operates for its own purpose, (N) entities distributed among the interconnected Open Systems work collectively to provide the (N) *service* to $(N+1)$ entities as illustrated in Fig. 4. In other words, the (N) entities add value to the $(N-1)$ service they get from the $(N-1)$ layer and offer this value-added service, i.e., the (N) service to the $(N+1)$ entities.

Communication between the $(N+1)$ entities make exclusive use of the (N) services. In particular, direct communication between the $(N+1)$ entities in the same system, e.g., for sharing resources, is not visible from outside of the system and thus is not covered by the OSI Architecture. Entities in the lowest layer communicate through the Physical Media for OSI, which could be considered as forming the (O) layer of the OSI Architecture. Cooperation between the (N) entities is ruled by the (N) *protocols* which precisely define how the (N) entities work together using the $(N-1)$ services to perform the (N) functions which add value to the $(N-1)$ service in order to offer the (N) service to the $(N+1)$ entities.

The (N) services are offered to the $(N+1)$ entities at the (N) *service access points*, or (N) *SAP's* for short, which represent the logical interfaces between the (N) entities and the $(N+1)$ entities. An (N) SAP can be served by only one

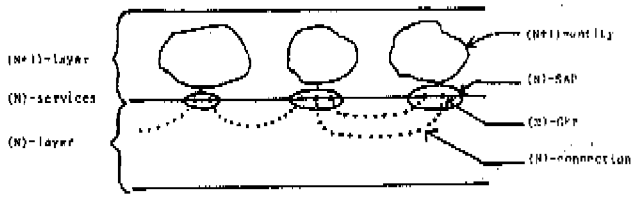


Fig. 5. Connections and connection endpoints (CEP's).

(*N*) entity and used by only one ($N + 1$) entity, but one (*N*) entity can serve several (*N*) SAP's and one ($N + 1$) entity can use several (*N*) SAP's.

A common service offered by all layers consists of providing associations between peer SAP's which can be used in particular to transfer data (it can, for instance, also be used to synchronize the served entities participating in the association). More precisely (see Fig. 5), the (*N*) layer offers (*N*) *connections* between (*N*) SAP's as part of the (*N*) services. The most usual type of connection is the *point-to-point* connection, but there are also *multiendpoint* connections which correspond to multiple associations between entities (e.g., broadcast communication). The end of an (*N*) connection at an (*N*) SAP is called an (*N*) *connection endpoint* or (*N*) *CEP* for short. Several connections may coexist between the same pair (or *n*-tuple) of SAP's.

Note: In the following, for the sake of simplicity, we will consider only point-to-point connections.

III. IDENTIFIERS

Objects within a layer or at the boundary between adjacent layers need to be uniquely identifiable, e.g., in order to establish a connection between two SAP's, one must be able to identify them uniquely. The OSI Architecture defines identifiers for entities, SAP's, and connections as well as relations between these identifiers, as briefly outlined below.

Each (*N*) entity is identified with a *global title*[1] which is unique and identifies the same (*N*) entity from anywhere in the network of Open Systems. Within more limited *domains*, an (*N*) entity can be identified with a *local title* which uniquely identifies the (*N*) entity only in the domain. For instance, within the domain corresponding to the (*N*) layer, (*N*) entities are identified with (*N*) *global titles* which are unique within the (*N*) layer.

Each (*N*) SAP is identified with an (*N*) *address* which uniquely identifies the (*N*)-SAP at the boundary between the (*N*) layer and the ($N + 1$) layer.

The concepts of titles and addresses are illustrated in Fig. 6.

Binding between (*N*) entities and the ($N - 1$) SAP's they use (i.e., SAP's through which they can access each other and communicate) are translated into the concept of (*N*) directory which indicates correspondence between global titles of (*N*) entities and (*N*) addresses through which they can be reached, as illustrated in Fig. 7.

[1] The term "title" has been preferred to the term "name" which is viewed as bearing a more general meaning. A title is equivalent to an entity name.

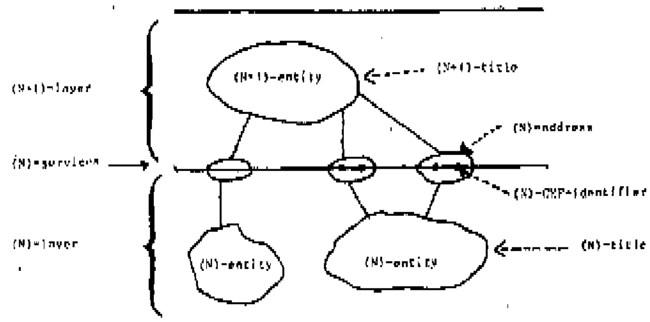


Fig. 6. Titles, addresses, and CEP-identifiers.

(N)-title	(N-1)-address
A	352
B	237
B	015
C	015

Fig. 7. Example of an (*N*)-directory.

Correspondence between (*N*) addresses served by an (*N*) entity and the ($N - 1$) addresses used for this purpose is performed by an (*N*) mapping function. In addition to the simplest case of one-to-one mapping, mapping may, in particular, be hierarchical with the (*N*) address being made of an ($N - 1$) address and an (*N*) suffix. Mapping may also be performed "by table." Those three types of mapping are illustrated in Fig. 8.

Each (*N*) CEP is uniquely identified within its (*N*) SAP by an (*N*) *CEP identifier* which is used by the (*N*) entity and the ($N + 1$) entity on both sides of the (*N*) SAP to identify the (*N*) connection as illustrated in Fig. 6. This is necessary since several (*N*) connections may end at the same (*N*) SAP.

IV. OPERATION OF CONNECTIONS

A. Establishment and Release

When an ($N + 1$) entity requests the establishment of an (*N*) connection from one of the (*N*) SAP's it uses to another (*N*) SAP, it must provide at the local (*N*) SAP the (*N*) address of the distant (*N*) SAP. When the (*N*) connection is established, both the ($N + 1$) entity and the (*N*) entity will use the (*N*)CEP identifier to designate the (*N*) connection.

(*N*) connections may be established and released dynamically on top of ($N - 1$) connections. Establishment of an (*N*) connection implies the availability of an ($N - 1$) connection between the two entities. If not available, the ($N - 1$) connection must be established. This requires the availability of an ($N - 2$) connection. The same consideration applies downwards until an available connection is encountered.

In some cases, the (*N*) connection may be established simultaneously with its supporting ($N - 1$) connection provided

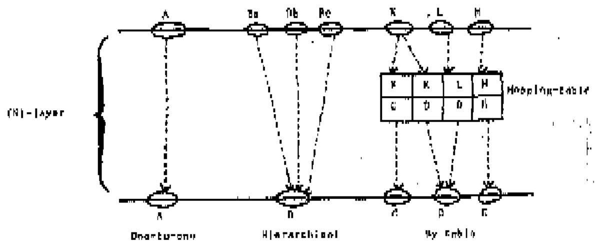


Fig. 8. Mapping between addresses.

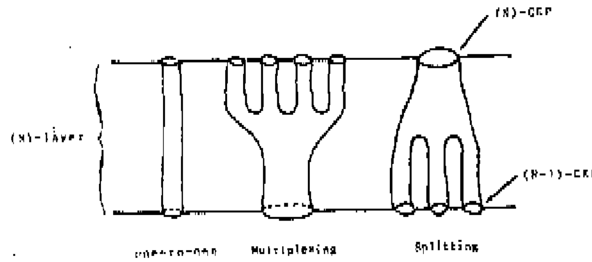


Fig. 9. Correspondence between connections.

the $(N-1)$ connection establishment service permits (N) entities to exchange information necessary to establish the (N) connection.

B. Multiplexing and Splitting

Three particular types of construction of (N) connections on top of $(N-1)$ connections are distinguished.

1) One-to-one correspondence where each (N) connection is built on one $(N-1)$ connection.

2) Multiplexing (referred to as "upward multiplexing" in [7]) where several (N) connections are multiplexed on one single $(N-1)$ connection.

3) Splitting (referred to as "downward multiplexing" in [7]) where one single (N) connection is built on top of several $(N-1)$ connection, the traffic on the (N) connection being divided between the various $(N-1)$ connections.

These three types of correspondence between connections in adjacent layers are illustrated in Fig. 9.

C. Data Transfer

Information is transferred in various types of data units between peer entities and between entities attached to a specific service access point. The data units are defined below and the interrelationship among several of them is illustrated in Fig. 10.

(N) *Protocol Control Information* is information exchanged between two (N) entities, using an $(N-1)$ connection, to coordinate their joint operation.

(N) *User Data* is the data transferred between two (N) entities on behalf of the $(N+1)$ entities for whom the (N) entities are providing services.

An (N) *Protocol Data Unit* is a unit of data which contains (N) Protocol Control Information and possibly (N) User Data.

(N) *Interface Control Information* is information exchanged between an $(N+1)$ entity and an (N) entity to coordinate their joint operation.

(N) *Interface Data* is information transferred from an $(N+1)$ entity to an (N) entity for transmission to a correspondent $(N+1)$ entity over an (N) connection, or conversely, information transferred from an (N) entity to an $(N+1)$ entity which has been received over an (N) connection from a correspondent $(N+1)$ entity.

(N) *Interface Data Unit* is the unit of information transferred across the service access point between an $(N+1)$ entity and an (N) entity in a single interaction. The size of (N) interface data units is not necessarily the same at each end of the connection.

	Control	Data	Combined
(N) – (N) Peer Entities	(N)-Protocol-Control-Information	(N)-User-Data	(N)-Protocol-Data-Units
(N)-(N-1) Adjacent layers	(N-1)-Interface-Control-Information	(N-1)-Interface Data	(N-1)-Interface Data-Unit

Fig. 10. Interrelationship between data units.

$(N-1)$ *Service Data Unit* is the amount of $(N-1)$ interface data whose identity is preserved from one end of an $(N-1)$ connection to the other. Data may be held within a connection until a complete service data unit is put into the connection.

Expedited $(N-1)$ service data unit is a small $(N-1)$ service data unit whose transfer is expedited. The $(N-1)$ layer ensures that an expedited data unit will not be delivered after any subsequent service data unit or expedited data unit sent on that connection. An expedited $(N-1)$ service data unit may also be referred to as an $(N-1)$ expedited data unit.

Note: An (N) protocol data unit may be mapped one-to-one onto an $(N-1)$ service data unit (see Fig. 11).

V. MANAGEMENT ASPECTS

Even though a number of resources are managed locally, i.e., without involving cooperation between distinct systems, some management functions do.

Examples of such management functions are

configuration information,
cold start/termination,
monitoring,
diagnostics,
reconfiguration, etc.

The OSI Architecture considers management functions as applications of a specific type. Management entities located in the highest layer of the architecture may use the complete set of services offered to all applications in order to perform

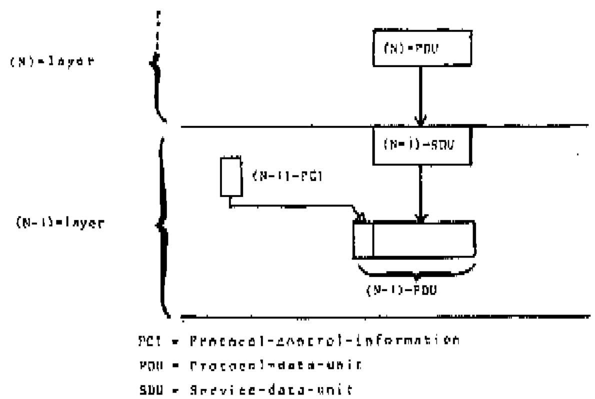


Fig. 11. Logical relationship between data units in adjacent layers.

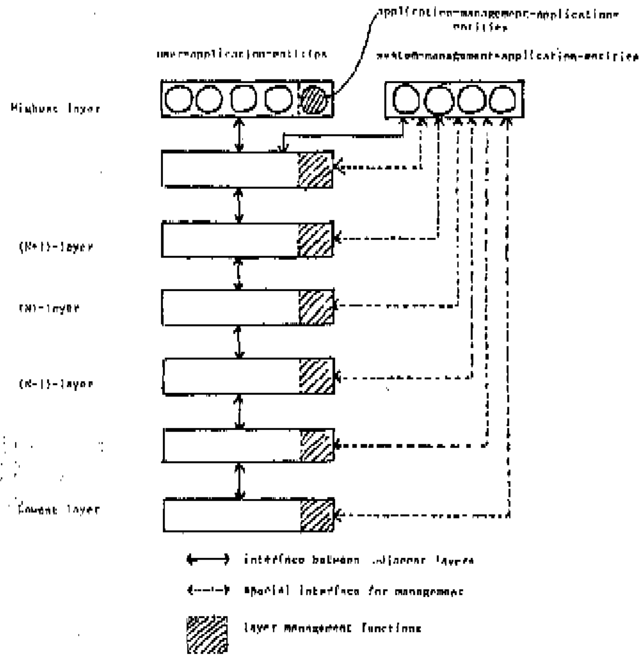


Fig. 12. A representation of management functions.

management functions. This organization of management functions within the OSI Architecture is illustrated in Fig. 12.

VI. THE SEVEN LAYERS OF THE OSI ARCHITECTURE

A. Justification of the Seven Layers

ISO determined a number of principles to be considered for defining the specific set of layers in the OSI architecture, and applied those principles to come up with the seven layers of the OSI Architecture.

Principles to be considered are as follows.

1) Do not create so many layers as to make difficult the system engineering task describing and integrating these layers.

2) Create a boundary at a point where the services description can be small and the number of interactions across the boundary is minimized.

3) Create separate layers to handle functions which are manifestly different in the process performed or the technology involved.

4) Collect similar functions into the same layer.

5) Select boundaries at a point which past experience has demonstrated to be successful.

6) Create a layer of easily localized functions so that the layer could be totaly redesigned and its protocols changed in a major way to take advantages of new advances in architectural, hardware, or software technology without changing the services and interfaces with the adjacent layers.

7) Create a boundary where it may be useful at some point in time to have the corresponding interface standardized.

8) Create a layer when there is a need for a different level of abstraction in the handling of data, e.g., morphology, syntax, semantics.

9) Enable changes of functions or protocols within a layer without affecting the other layers.

10) Create for each layer interfaces with its upper and lower layer only.

11) Create further subgrouping and organization of functions to form sublayers within a layer in cases where distinct communication services need it.

12) Create, where needed, two or more sublayers with a common, and therefore minimum, functionality to allow interface operation with adjacent layers.

13) Allow bypassing of sublayers.

B. Specific Layers

The following is a brief explanation of how the layers were chosen.

1) It is essential that the architecture permits usage of a realistic variety of physical media for interconnection with different control procedures (e.g., V.24, V.35, X.21, etc.). Application of principles 3, 5, and 8 leads to identification of a *Physical Layer* as the lowest layer in the architecture.

2) Some physical communications media (e.g., telephone line) require specific techniques to be used in order to transmit data between systems despite a relatively high error rate (i.e., an error rate not acceptable for the great majority of applications). These specific techniques are used in data-link control procedures which have been studied and standardized for a number of years. It must also be recognized that new physical communications media (e.g., fiber optics) will require different data-link control procedures. Application of principles 3, 5, and 8 leads to identification of a *Data link Layer* on top of the Physical Layer in the architecture.

3) In the Open Systems Architecture, some systems will act as final destination of data. Some systems may act only as intermediate nodes (forwarding data to other systems). Application of principles 3, 5, and 7 leads to identification of a *Network Layer* on top of the Data link Layer. Network-oriented protocols such as routing, for example, will be grouped in this layer. Thus, the Network Layer will provide a connection path (network connection) between a pair of transport entities (see Fig. 13).

4) Control of data transportation from source end system to destination end system (which need not be performed in

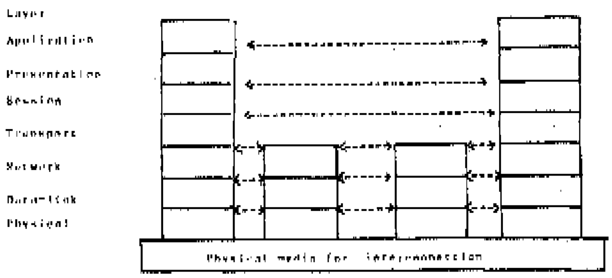


Fig. 13. The seven layers OSI architecture.

intermediate nodes) is the last function to be performed in order to provide the totality of the transport service. Thus, the upper layer in the transport-service part of the architecture is the *Transport Layer,* sitting on top of the Network Layer. This Transport Layer relieves higher layer entities from any concern with the transportation of data between them.

5) In order to bind/unbind distributed activities into a logical relationship that controls the data exchange with respect to synchronization and structure, the need for a dedicated layer has been identified. So the application of principles 3 and 4 leads to the establishment of the *Session Layer* which is on top of the Transport Layer.

6) The remaining set of general interest functions are those related to representation and manipulation of structured data for the benefit of application programs. Application of principles 3 and 4 leads to identification of a *Presentation Layer* on top of the Session Layer.

7) Finally, there are applications consisting of application processes which perform information processing. A portion of these application processes and the protocols by which they communicate comprise the *Application Layer* as the highest layer of the architecture.

The resulting architecture with seven layers, illustrated in Fig. 13 obeys principles 1 and 2.

A more detailed definition of each of the seven layers identified above is given in the following sections, starting from the top with the application layer described in Section VI-C1) down to the physical layer described in Section VI-C7).

C. Overview of the Seven Layers of the OSI Architecture

1) *The Application Layer*: This is the highest layer in the OSI Architecture. Protocols of this layer directly serve the end user by providing the distributed information service appropriate to an application, to its management, and to system management. Management of Open Systems Interconnection comprises those functions required to initiate, maintain, terminate, and record data concerning the establishment of connections for data transfer among application processes. The other layers exist only to support this layer.

An application is composed of cooperating *application processes* which intercommunicate according to application layer protocols. Application processes are the ultimate source and sink for data exchanged.

A portion of an application process is manifested in the application layer as the execution of application protocol (i.e., application entity). The rest of the application process is considered beyond the scope of the present layered model. Applications or application processes may be of any kind (manual, computerized, industrial, or physical).

2) *The Presentation Layer*: The purpose of the Presentation Layer is to provide the set of services which may be selected by the Application Layer to enable it to interpret the meaning of the data exchanged. These services are for the management of the entry exchange, display, and control of structured data.

The presentation service is location-independent and is considered to be on top of the Session Layer which provides the service of linking a pair of presentation entities.

It is through the use of services provided by the Presentation Layer that applications in an Open Systems Interconnection environment can communicate without unacceptable costs in interface variability, transformations, or application modification.

3) *The Session Layer*: The purpose of the Session Layer is to assist in the support of the interactions between cooperating presentation entities. To do this, the Session Layer provides services which are classified into the following two categories.

a) Binding two presentation entities into a relationship and unbinding them. This is called *session administration service.*

b) Control of data exchange, delimiting, and synchronizing data operations between two presentation entities. This is called *session dialogue service.*

To implement the transfer of data between presentation entities, the Session Layer may employ the services provided by the Transport Layer.

4) *The Transport Layer*: The Transport Layer exists to provide a universal transport service in association with the underlying services provided by lower layers.

The Transport Layer provides transparent transfer of data between session entities. The Transport Layer relieves these session entities from any concern with the detailed way in which reliable and cost-effective transfer of data is achieved.

The Transport Layer is required to optimize the use of available communications services to provide the performance required for each connection between session entities at a minimum cost.

5) *The Network Layer*: The Network Layer provides functional and procedural means to exchange network service data units between two transport entities over a network connection. It provides transport entities with independence from routing and switching considerations.

6) *The Data Link Layer*: The purpose of the Data link Layer is to provide the functional and procedural means to establish, maintain, and release data links between network entities.

7) *The Physical Layer*: The Physical Layer provides mechanical, electrical, functional, and procedural characteristics to establish, maintain, and release physical connections (e.g., data circuits) between data link entities.

VII. OSI PROTOCOLS DEVELOPMENTS

The model of OSI Architecture defines the services provided by each layer to the next higher layer, and offers con-

cepts to be used to specify how each layer performs its specific functions.

Detailed functioning of each layer is defined by the protocols specific to the layer in the framework of the Architecture model.

Most of the initial effort within ISO has been placed on the model of OSI. The next step consists of the definition of standard protocols for each layer.

This section contains a brief description of a likely initial set of protocols, corresponding to specific standardization projects recommended by SC16.

A. Protocols in the Physical Layer

Standards already exist within CCITT defining:

1) interfaces with physical media for OSI, and

2) protocols for establishing, controlling, and releasing switched data circuits.

Such standards are described in other papers in this issue [9], [10], e.g., X.21, V.24, V.35, etc.

The only work to be done will consist of clearly relating those standards to the OSI Architecture model.

B. Protocols in the Data Link Layer

Standard protocols for the Data link Layer have already been developed within ISO, which are described in other papers within this issue [11], [12].

The most popular Data link Layer protocol is likely to be HDLC [13], without ruling out the possibility of using also other character-oriented standards.

Just as for the Physical Layer, the remaining work will consist mainly of clearly relating these existing standards to the OSI Architecture model.

C. Protocols in the Network Layer

An important basis for protocols in the network layer is level 3 of the X.25 interface [14] defined by CCITT and described in another paper in this issue. It will have to be enhanced in particular to permit interconnection of private and public networks.

Other types of protocols are likely to be standardized later in this layer, and in particular, protocols corresponding to Datagram networks [10].

D. Protocols in the Transport Layer

No standard exists at present for this layer; a large amount of experience has been accumulated in this area and several proposals are available.

The most widely known proposal is the Transport Protocol proposed by IFIP and known as INWG 96.1 [15], which could serve as a basis for defining an international standard.

E. Protocols for the Session Layer

No standard exists and no proposal has been currently available, since in most networks, session functions were often considered as part of higher layer functions such as Virtual Terminal and File Transfer.

A standard Session Layer Protocol can easily be extracted from existing higher layer protocols.

F. Presentation Layer Protocol

So far, Virtual Terminal Protocols and part of Virtual File are considered the most urgent protocols to be developed in the Presentation Layer.

A number of VTP's are available (e.g., [16], [17]), many of them being very similar, and it should be easy to derive a Standard VTP from these proposals, also making use of the ISO standard for "Extended Control Characters for I/O Imaging Devices" [18]. These protocols are reviewed in another paper in this issue [19].

The situation is similar for File Transfer Protocols.

G. Management Protocols

Most of the work within ISO has been done so far on the architecture of management functions, and very little work has been done on management protocols themselves. Therefore, it is too early to give indications on the likely results of the ISO work in this area.

VIII. CONCLUSION

The development of OSI Standards is a very big challenge,, the result of which will impact all future computer communication developments. If standards come too late or are inadequate, interconnection of heterogeneous systems will not be possible or will be very costly.

The work collectively achieved so far by SC16 members is very promising, and additional efforts should be expended to capitalize on these initial results and come up rapidly with the most urgently needed set of standards which will support initial usage of OSI (mainly terminals accessing services and file transfers). The next set of standards, including OSI management and access to distributed data, will have to follow very soon.

Common standards between ISO and CCITT are also essential to the success of standardization, since new services announced by PTT's and common carriers are very similar to data processing services offered as computer manufacturer products, and duplication of now compatible standards could simply cause the standardization effort to fail. In this regard, acceptance of the OSI Reference Model by CCITT Rapporteur's Group on Layered Architecture for Public Data Networks Services is most promising.

It is essential that all partners in this standardization process expend their best effort so it will be successful, and the benefits can be shared by all users, manufacturers of terminals and computers, and the PTT's/common carriers.

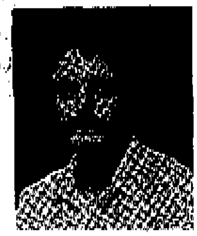
ACKNOWLEDGMENT

The OSI Architecture model briefly described in this paper results from the work of more than 100 experts from many countries and international organizations. Participation in this collective work was really a fascinating experience for the author who acknowledges the numerous contributions from SC16 members which have been merged in the final version of the OSI Architecture briefly presented here.

REFERENCES

[1] L. G. Roberts and B. D. Wessler, "Computer network development to achieve resource sharing," in *Proc. SJCC*, 1970, pp. 543–549.
[2] L. Pouzin, "Presentation and major design aspects of the CYCLADES computer network," in *Proc. 3rd ACM-IEEE Commun. Symp.*, Tampa, FL, Nov. 1973, pp. 80–87.
[3] J. H. McFayden, "Systems network architecture: An overview," *IBM Syst. J.*, vol. 15, no. 1, pp. 4–23, 1976.
[4] G. E. Conant and S. Wecker, "DNA, An Architecture for heterogeneous computer networks," in *Proc., ICCC*, Toronto, Ont., Canada, Aug. 1976, pp. 618–625.
[5] H. Zimmermann, "High level protocols standardization: Technical and political issues," in *Proc. ICCC*, Toronto, Ont., Canada, Aug. 1976, pp. 373–376.
[6] ISO/TC97/SC16, "Provisional model of open systems architecture," Doc. N34, Mar. 1978.
[7] ISO/TC97/SC16, "Reference model of open systems interconnection," Doc. N227, June 1979.
[8] H. Zimmermann and N. Naffah, "On open systems architecture," in *Proc. ICCC*, Kyoto, Japan, Sept. 1978, pp. 669–674.
[9] H. V. Bertine, "Physical level protocols," this issue pp. 433–444.
[10] H. C. Folts, "Procedures for circuit-switched service in synchronous public data networks," and "X.25 transaction-oriented features—Datagram and fast select," this issue, pp. 489–496.
[11] J. W. Conard, "Character oriented data link control protocols," this issue, pp. 445–454.
[12] D. E. Carlson, "Bit-oriented data link control procedures," this issue, pp. 455–467.
[13] ISO, "High level data link control-elements of procedure," IS 4335, 1977.
[14] CCITT, "X25," *Orange Book*, vol. VIII-2, 1977, pp. 70–108.
[15] IFIP-WG 6.1, "Proposal for an internetwork end-to-end transport protocol," INWG Note 96.1; also, doc. ISO/TC97 SC16/N24, 46 pp., Mar. 1978.
[16] IFIP-WG 6.1, "Proposal for a standard virtual terminal protocol," doc. ISO/TC97/SC16/N23, 56 pp., Feb. 1978.
[17] EURONET, "Data entry virtual terminal protocol for EURONET," VTP/D-Issue 4, doc. EEC/WGS/165.
[18] ISO, "Extended control characters for I/0 imaging devices," DP 6429.
[19] J. Day, "Terminal protocols," this issue, pp. 585–593.

★



Hubert Zimmermann received degrees in engineering from École Polytechnique, Paris, France, in 1963, and from Ecole Nationale Superieure des Telecommunications, Paris, France, in 1966.

He is presently in charge of the computer communications group at IRIA, Rocquencourt, France. He was involved in development of command and control systems before joining IRIA in 1972 to start the CYCLADES project with L. Pouzin. Within CYCLADES, he was mainly responsible for design and implementation of host protocols.

Dr. Zimmermann is a member of IFIP WG 6.1 [International Network Working Group (INWG)]. He also chaired the Protocol Subgroup and co-authored several proposals for international protocols. He is an active participant in the development of standards for Open Systems Interconnection (OSI) within ISO, where he chairs the working group on OSI architecture.

OSI model

From Wikipedia, the free encyclopedia

OSI Model
7 Application layer
6 Presentation layer
5 Session layer
4 Transport layer
3 Network layer
2 Data link layer ▪ LLC sublayer ▪ MAC sublayer
1 Physical layer

The **Open Systems Interconnection Basic Reference Model** (*OSI Reference Model* or *OSI Model* for short) is a layered, abstract description for communications and computer network protocol design, developed as part of Open Systems Interconnection (OSI) initiative. It is also called the **OSI seven layer model**. The layers, described below, are, from top to bottom, Application, Presentation, Session, Transport, Network, Data Link and Physical.

Even though newer IETF and IEEE protocols, and indeed OSI protocol work subsequent to the publication of the original architectural standards have largely superseded it, the OSI model is an excellent place to begin the study of network architecture. Not understanding that the pure seven-layer model is more historic than current, many beginners make the mistake of trying to fit every protocol they study into one of the seven basic layers. This is not always easy to do as many of the protocols in use on the Internet today were designed as part of the TCP/IP model, and may not fit cleanly into the OSI model.

Contents

History

In 1977, the International Organization for Standardization (ISO), began to develop its OSI networking suite. OSI has two major components: an abstract model of networking (the Basic Reference Model, or seven-layer model), and a set of concrete protocols. The standard documents that describe OSI are for sale and not currently available online.

Parts of OSI have influenced Internet protocol development, but none more than the abstract model itself, documented in ISO 7498 and its various addenda. In this model, a networking system is divided into layers. Within each layer, one or more entities implement its functionality. Each entity interacts directly only with the layer immediately beneath it, and provides facilities for use by the layer above it.

In particular, Internet protocols are deliberately not as rigorously architected as the OSI model, but a common version of the TCP/IP model splits it into four layers. The Internet Application Layer includes the OSI Application Layer, Presentation Layer, and most of the Session Layer. Its End-to-End Layer includes the graceful close function of the OSI Session Layer as well as the Transport Layer. Its Internetwork Layer is equivalent to the OSI Network Layer, while its Interface layer includes the OSI Data Link and Physical Layers. These comparisons are based on the original seven-layer protocol model as defined in ISO 7498, rather than refinements in such things as the Internal Organization of the Network Layer document.

Protocols enable an entity in one host to interact with a corresponding entity at the same layer in a remote host. Service definitions abstractly describe the functionality provided to a (N)-layer by an (N-1) layer, where N is one of the seven layers inside the local host.

Description of OSI layers

Layer 7: Application layer

OSI Model			
	Data unit	**Layer**	**Function**
Host layers	Data	Application	Network process to application
		Presentation	Data representation and encryption
		Session	Interhost communication
	Segments	Transport	End-to-end connections and reliability (TCP)
Media layers	Packets	Network	Path determination and logical addressing (IP)
	Frames	Data link	Physical addressing (MAC & LLC)
	Bits	Physical	Media, signal and binary transmission

The application layer is the seventh level of the seven-layer OSI model. It interfaces directly to and performs common application services for the application processes; it also issues requests to the presentation layer. Note carefully that this layer provides services to user-defined application processes, and not to the end user. For example, it defines a file transfer protocol, but the end user must go through an application process to invoke file transfer. The OSI model does not include human interfaces.

The common application services sublayer provides functional elements including the Remote Operations Service Element (comparable to Internet Remote Procedure Call), Association Control, and Transaction Processing (according to the ACID requirements).

Above the common application service sublayer are functions meaningful to user application programs, such as messaging (X.400), directory (X.500), file transfer (FTAM), virtual terminal (VTAM), and batch job manipulation (JTAM).

Layer 6: Presentation layer

The Presentation layer transforms the data to provide a standard interface for the Application layer. MIME encoding, data encryption and similar manipulation of the presentation are done at this layer to present the data as a service or protocol that the developer sees fit. Examples of this layer are converting an EBCDIC-coded text file to an ASCII-coded file, or serializing objects and other data structures into and out of XML.

Layer 5: Session layer

The Session layer controls the dialogues/connections (sessions) between computers. It establishes, manages and terminates the connections between the local and remote application. It provides for either full-duplex or half-duplex operation, and establishes checkpointing, adjournment, termination, and restart procedures. The OSI model made this layer responsible for "graceful close" of sessions, which is a property of TCP, and also for session checkpointing and recovery, which is not usually used in the Internet protocols suite. Session layers are commonly used in application environments that make use of remote procedure calls (RPCs).

iSCSI, which implements the Small Computer Systems Interface (SCSI) encapsulated into TCP/IP packets, is a session layer protocol increasingly used in Storage Area Networks and internally between processors and high-performance storage devices. iSCSI leverages TCP for guaranteed delivery, and carries SCSI command descriptor blocks (CDB) as payload to create a virtual SCSI bus between iSCSI initiators and iSCSI targets.

Layer 4: Transport layer

The Transport layer provides transparent transfer of data between end users, providing reliable data transfer services to the upper layers. The transport layer controls the reliability of a given link through flow control, segmentation/desegmentation, and error control. Some protocols are state and connection oriented. This means that the transport layer can keep track of the segments and retransmit those that fail. The best known example of a layer 4 protocol is the Transmission Control Protocol (TCP). The transport layer is the layer that converts messages into TCP segments or User Datagram Protocol (UDP), Stream Control Transmission Protocol (SCTP), etc. packets. Perhaps an easy way to visualize the Transport Layer is to compare it with a Post Office, which deals with the dispatch and classification of mail and parcels sent. Do remember, however, that a post office manages the outer envelope of mail. Higher layers may have the equivalent of double envelopes, such as cryptographic Presentation services that can be read by the addressee only. Roughly speaking, tunneling protocols operate at the transport layer, such as carrying non-IP protocols such as IBM's SNA or Novell's IPX over an IP network, or end-to-end encryption with IPsec. While Generic Routing Encapsulation (GRE) might seem to be a network layer protocol, if the encapsulation of the payload takes place only at endpoint, GRE becomes closer to a transport protocol that uses IP headers but contains complete frames or packets to deliver to an endpoint. L2TP carries PPP frames inside transport packets.

Layer 3: Network layer

The Network layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination via one or more networks while maintaining the quality of service requested by the Transport layer. The Network layer performs network routing functions, and might also perform fragmentation and reassembly, and report delivery errors. Routers operate at this layer—sending data throughout the extended network and making the Internet possible. This is a logical addressing scheme – values are chosen by the network engineer. The addressing scheme is hierarchical. The best known example of a layer 3 protocol is the Internet Protocol (IP). Perhaps it's easier to visualize this layer as managing the sequence of human carriers taking a letter from the sender to the local post office, trucks that carry sacks of mail to other post offices or airports, airplanes that carry airmail between major cities, trucks that distribute mail sacks in a city, and carriers that take a letter to its destinations. Think of fragmentation as splitting a large document into smaller envelopes for shipping, or, in the case of the network layer, splitting an application or transport record into packets.

Layer 2: Data Link layer

The Data Link layer provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the Physical layer. The best known example of this is Ethernet. This layer manages the interaction of devices with a shared medium. Other examples of data link protocols are HDLC and ADCCP for point-to-point or packet-switched networks and Aloha for local area networks. On IEEE 802 local area networks, and some non-IEEE 802 networks such as FDDI, this layer may be split into a Media Access Control (MAC) layer and the IEEE 802.2 Logical Link Control (LLC) layer. It arranges bits from the physical layer into logical chunks of data, known as frames.

This is the layer at which the bridges and switches operate. Connectivity is provided only among locally attached network nodes forming layer 2 domains for unicast or broadcast forwarding. Other protocols may be imposed on the data frames to create tunnels and logically separated layer 2 forwarding domain.

The data link layer might implement a sliding window flow control and acknowledgment mechanism to provide reliable delivery of frames; that is the case for SDLC and HDLC, and derivatives of HDLC such as LAPB and LAPD. In modern practice, only error detection, not flow control using sliding window, is present in modern data link protocols such as Point-to-Point Protocol (PPP), and, on local area networks, the IEEE 802.2 LLC layer is not used for most protocols on Ethernet, and, on other local area networks, its flow control and acknowledgment mechanisms are rarely used. Sliding window flow control and acknowledgment is used at the transport layers by protocols such as TCP.

Layer 1: Physical layer

The Physical layer defines all the electrical and physical specifications for devices. In particular, it defines the relationship between a device and a physical medium. This includes the layout of pins, voltages, and cable specifications. Hubs, repeaters, network adapters and Host Bus Adapters (HBAs used in Storage Area Networks) are physical-layer devices.

To understand the function of the physical layer in contrast to the functions of the data link layer, think of the physical layer as concerned primarily with the interaction of a single device with a medium, where the data link layer is concerned more with the interactions of multiple devices (i.e., at least two) with a shared medium. The physical layer will tell one device how to transmit to the medium, and another device how to receive from it, but not, with modern protocols, how to gain access to the medium. Obsolescent physical layer standards such as RS-232 do use physical wires to control access to the medium.

The major functions and services performed by the physical layer are:

- Establishment and termination of a connection to a communications medium.
- Participation in the process whereby the communication resources are effectively shared among multiple users. For example, contention resolution and flow control.
- Modulation, or conversion between the representation of digital data in user equipment and the corresponding signals transmitted over a communications channel. These are signals operating over the physical cabling (such as copper and optical fiber) or over a radio link.

Parallel SCSI buses operate in this layer, although it must be remembered that the logical SCSI protocol is a transport-layer protocol that runs over this bus.. Various physical-layer Ethernet standards are also in this layer; Ethernet incorporates both this layer and the data-link layer. The same applies to other local-area networks, such as Token ring, FDDI, and IEEE 802.11, as well as personal area networks such as Bluetooth and IEEE 802.15.4.

Mnemonics

- **P**lease **D**o **N**ot **T**hrow **S**alami **P**izza **A**way. — this works for bottom-to-top (layer 1 to 7).
- **A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing. — a top-to-bottom reminder (layer 7 to 1).
- **APS** **T**ransports **N**etwork **D**ata **P**hysically. — APS refers to *Application*, *Presentation*, and *Session*. This one separates the upper and lower layer groups.
- **P**lease **D**o **N**ot **T**ell **S**ecret **P**asswords **A**nytime. — Another bottom-to-top phrase.

Interfaces

In addition to standards for individual protocols in transmission, there are also interface standards for different layers to talk to the ones above or below (usually operating-system–specific). For example, Microsoft Windows' Winsock, and Unix's Berkeley sockets and System V Transport Layer Interface, are interfaces between applications (layers 5 and above) and the transport (layer 4). NDIS and ODI are interfaces between the media (layer 2) and the network protocol (layer 3).

OSI Service Specifications are abstractions of functionality commonly present in programming interfaces.

Examples

Layer #	Layer Name	Misc. examples	TCP/IP suite	SS7	AppleTalk suite	OSI suite	IPX suite	SNA	UMTS
7	Application	NNTP, HL7, Modbus, SIP,	DHCP, DNS, FTP, Gopher, HTTP, NFS, NTP, RTP, SMPP,	ISUP, INAP, MAP,	AFP	FTAM, X.400, X.500,		APPC	

		SSI	SMTP, SNMP, Telnet	TUP, TCAP		DAP			
6	Presentation	TDI, ASCII, EBCDIC, MIDI, MPEG	MIME, XDR, SSL, TLS (Not a separate layer)		AFP	ISO 8823, X.226			
5	Session	Named Pipes, NetBIOS, SAP, SDP	Sockets. Session establishment in TCP. SIP. (Not a separate layer with standardized API.)		ASP, ADSP, ZIP, PAP	ISO 8327, X.225	NWLink	DLC?	
4	Transport	NBF, nanoTCP, nanoUDP	TCP, UDP, SCTP		ATP, NBP, AEP, RTMP	TP0, TP1, TP2, TP3, TP4	SPX		
3	Network	NBF, Q.931	IP, ICMP, IPsec, ARP, RIP, OSPF	MTP-3, SCCP	DDP	X.25 (PLP), CLNP	IPX		RRC (Radio Resource Control)
2	Data Link	802.3 (Ethernet), 802.11a/b/g/n MAC/LLC, 802.1Q (VLAN), ATM, CDP, FDDI, Fibre Channel, Frame Relay, HDLC, ISL, PPP, Q.921, Token Ring	PPP, SLIP, PPTP, L2TP	MTP-2	LocalTalk, TokenTalk, EtherTalk, AppleTalk Remote Access, PPP	X.25 (LAPB), Token Bus	IEEE 802.3 framing, Ethernet II framing	SDLC	RLC (Radio Link Control), MAC (Media Access Control), PDCP (Packet Data Convergence Protocol) and Broadcast/Multicast Control (BMC).
1	Physical	RS-232, V.35, V.34, I.430, I.431, T1, E1, 10BASE-T, 100BASE-TX, POTS, SONET, DSL, 802.11a/b/g/n PHY		MTP-1	RS-232, RS-422, STP, PhoneNet	X.25 (X.21bis, EIA/TIA-232, EIA/TIA-449, EIA-530, G.703)		Twinax	UMTS L1 (UMTS Physical Layer)

See also

- TCP/IP model
- Internet protocol suite
- List of network protocols

- OSI protocols
- Node

External links

- ISO standard 7498-1:1994 (ZIP format)
- Cybertelecom — Layered Model of Regulation
- OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection PDF (776 KiB), Hubert Zimmermann, IEEE Transactions on Communications, vol. 28, no. 4, April 1980, pp. 425 - 432.
- Internetworking Basics
- osi7layer.com Amusing list of OSI mnemonics

Retrieved from "http://en.wikipedia.org/wiki/OSI_model"

Categories: CISSP | Network architecture | ISO standards | ITU-T recommendations | OSI protocols | Reference models

APPENDIX III: RELATED PROCEEDINGS

None